04

Otros métodos usando Collections y Streams

Transcripción

[00:00] Hola. ¿Cómo están? En nuestra clase número 10, última de nuestro bloque conociendo más de listas, vamos a aprender más de streams. Para ello por ejemplo vamos a hacer varios ejemplos utilizando streams. Para eso vamos a utilizar aquí nuestra clase 9, copiar la clase 10. Una vez hecho esto, vamos a darle aquí adicionar unos cursos más.

[00:36] Curso4, curso5, curso6, curso7, curso8. Geometría, física, química, geografía y por ejemplo vamos a colocar aquí educación física. Vamos a tener 10, 20, 30, 60, 80, 70 y 30. De nuevo. Una vez hecho esto, aquí vamos a adicionar nuestros cursos nuevos, 5, 6, 7 y 8. Curso5, curso6, curso7, curso8. Perfecto.

[01:53] Vamos a mover aquí todo lo que es collections y vamos a mantenernos solamente en la parte de streams. Aquí por ejemplo estamos utilizando suma, ¿ahora tenemos cuánto? 320 horas. Ahora quitando, tenemos 320 horas, quitando prácticamente historia. Ahora vamos a remover ese filtro de historia y vamos a hacer lo siguiente.

[02:33] En lugar de suma, queremos el promedio. Utilizamos el average. Después por ejemplo queremos el número máximo, utilizamos el max. Queremos el número mínimo, utilizamos el min. ¿Quién tiene más tiempo? El promedio. En el caso del average, retorna aquí un optional, todos retornaron un optional. Aquí sería 43.75, 80 y 10. El max puedo ponerle aquí getAsInt, getAsInt y aquí getAsDouble.

[03:32] Tenemos aquí 320, 43.75, 80 y 10. Ahora, supongamos que tengamos cursos repetidos, cosa que no va a existir pero por ejemplo, vamos a colocar aquí, vamos a tener física. Vamos a tener dos veces física y dos veces química. Ahora nosotros queremos agrupar. ¿Qué sucede cuando queremos agrupar?

[04:07] Podemos decir group by, tipo como si fuese un SQL, pero aquí podemos hacer lo siguiente. Vamos a hacer aquí un map, creamos un mapa, vamos a colocar aquí string que sería prácticamente lo que queremos que duplique, queremos que sea duplicado. A su vez, vamos a colocar aquí una lista de curso. Y vamos a colocar aquí groupCurso, igual.

[05:02] ¿Aquí qué vamos a hacer? Vamos a colocar aquí post, cursos.stream().collect, vamos a colocar aquí collectors.groupingBy, vamos a colocar curso.getNombre. Estamos diciendo que hagas una lista, agrupa, y lo que está repetido coloca aquí, y aquí vas a colocar la lista repetido, la lista nueva.

[05:48] Una vez hecho esto, podemos aquí hacer un group.forEach simple y vamos a colocar aquí curso y vamos a imprimir, System.out.println y vamos a imprimir aquí curso. ¿Aquí qué sucede? Aquí vamos a tener que realizar dos cosas. Primero, tenemos un map de stream con una lista, para eso primero necesitamos, ¿qué cosa? El key y el value, que sería aquí, vamos a utilizar así. Key y value.

[06:40] Vamos a imprimir el key. El key sería lo repetido y el value sería la lista de lo que agrupó. System.out.prinln y el key, imprimimos el key. Ahora ejecutamos nuevamente. ¿Imprimió qué cosa? Álgebra, geometría, aritmética, química, física e historia. Ahora no quiero que sea esto de aquí, quiero que también imprima la cantidad de registros que agrupó.

[07:10] Entonces vamos a colocar aquí +, y aquí sería value.size. Una vez hecho esto, contamos nuevamente. Tenemos dos de física y dos de química, dos de química y dos de física repetidos. Todo eso utilizando stream. Ahora tenemos

un montón de opciones para utilizar stream, un montón de oportunidades para poder utilizar stream, pero además quiero hacer un punto adicional, para finalizar.

[07:57] También tenemos stream, y a su vez de stream también tenemos el parallelStream. ¿Qué cosa es el parallelStream? El parallelStream se va a encargar de hacer tipo un stream en threads. ¿Qué cosa? Cuando yo tengo en threads por ejemplo él va a separar el número de cursos, tenemos cuatro cursos, él va a separar en cuatro threads.

[08:21] Así supongamos que nosotros queramos sumar toda una cantidad, podemos obtener un resultado mucho más rápido que hacer un stream simple por ejemplo. Aquí tenemos parallelStream, punto, el count no necesitamos por ejemplo, punto mapToInt(Curso::getTiempo).sum();

[09:14] 350, quitándole claro, porque ahora hemos adicionado historia, que antes habíamos removido. ¿Qué hizo? Él separó en varias threads. ¿Para qué opción utilizar parallelStream y para cuál no? Voy a dar un ejemplo. Imaginemos que nosotros queremos el número máximo, y queremos el número máximo utilizando el parallelStream.

[09:39] Al tener cuatro threads, puede ser que el número máximo él tenga como 50, porque ese fue el último stream que obtuvo, y no 60, ahí podemos hacer un error. Igual con el mínimo. Si queremos un filtro con geometría o aritmética, va a ser más rápido, porque él va a separar en cuatro threads, como que va a dividir en cuatro bloques nuestra lista y va a procurar aritmética mucho más rápido.

[10:05] La búsqueda será mejor. Eso también tiene un recurso del CPU, el curso de los cores del servidor, para eso también es bueno saber en qué situaciones utilizarlo. Para eso también se tienen que hacer pruebas de tiempo y de performance. Esto básicamente sería nuestra clase 10. Estamos terminando

nuestro bloque conociendo más de listas y nos vemos en la siguiente clase.

Muchas gracias.